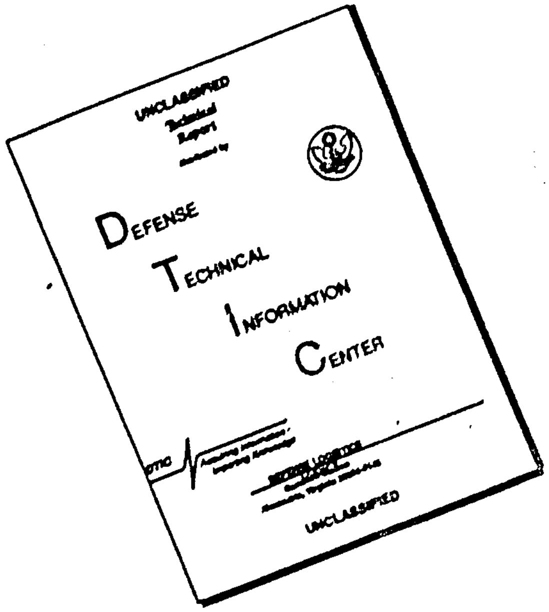
DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

64-11

Carnegie Institute of Technology

Pittsburgh 13, Pennsylvania

GRADUATE SCHOOL of INDUSTRIAL ADMINISTRATION

William Larimer Mellon, Founder

(14) Rept. no. ONR RM 112

~~O.N.R. Research Memorandum No. 112~~

(6) AN ALL-INTEGER INTEGER PROGRAMMING ALGORITHM*

(10) by

Fred Glover

(11) Dec~~ember,~~ ~~1~~963

Graduate School of Industrial Administration, Pittsburgh, Pa.

(5) 169200 Carnegie Inst~~itute~~ of Tech~~nology~~.

~~Pittsburgh, Pennsylvania 15213~~

* I would like to acknowledge the stimulus of a research association with Professor Gerald L. Thompson.

This paper was written as part of the contract "Planning and Control of Industrial Operations," with the Office of Naval Research at the Graduate School of Industrial Administration, Carnegie Institute of Technology, Management Sciences Research Group. Reproduction of this paper in whole or in part is permitted for any purpose of the United States Government. Contract Nonr 27 7024.

(15)

AI

Form of the Problem. Using matrix notation, the standard problem may be written

$$\text{(1)} \quad \text{Minimize } wb + b_0$$
$$\text{subject to} \quad wA^o \geq c^o$$
$$\text{and} \quad w \geq 0, \; w \text{ integer}$$

where b is an m X 1 column vector, b_0 is a scalar, c^o and 0 are 1Xn row vectors, A^o is an mXn rectangular matrix, and w is a 1Xn row vector whose values we wish to find so as to achieve the minimiaation objective and satisfy the constraining conditions of (1). The fact that w must be integral distinguishes the problem from the general linear programming problem which allows w to have fractional components.

For the purpose of this paper we define the mX(m + n) augmented matrix A and the 1X(m + n) augmented vector c by

$$A = (A^o \;\; I), \qquad c = (c^o \;\; 0),$$

(where I is the mXm identity matrix) and rewrite (1) as

$$\text{(2)} \quad \text{Minimize} \quad wb + b_0$$
$$\text{subject to} \quad wA \geq c, \; w \text{ integer}.$$

We assume without loss of generality that the augmented matrix (-b A) is lexicographically negative by row, for if it is not it may readily be made so (see [1]). Following Gomory's terminology [3], we will call our method an all-integer algorithm, for we additionally require that all elements of A, b, and c be integral, or at least commensurable. In practical terms this is of course no restriction since numbers are represented in the computer with finite decimal expansions in any case.

Tools of the Algorithm.

1. A set of transformations which will change the problem into a new problem in nonnegative variables so that any optimal integer solution

to the new problem provides an optimal integer solution to the original, and conversely. We will call these transformations elemental transformations.

2. A procedural rule (coupled with a rule of choice) for applying the elemental transformations in order to create a new problem containing a submatrix of a special form, which we will call the bounding form.

3. A process called the bound escalation method for operating on the bounding form to supply lower bound values for some set of the problem variables.

Thus the algorithm may be roughly sketched as follows.

1. Apply a series of elemental transformations to obtain an equivalent problem in new variables which exhibits a bounding form matrix B.

2. Apply the bound escalation method to B. At the end of the process the lower bound values assigned to a subset of the problem variables will satisfy all the constraints associated with B.

3. Adjust the c vector to reflect the assignment of lower bounds established in 2. If c becomes nonpositive, the problem is solved. Otherwise, return to 1 and repeat.

Several features of the algorithm may be noted. First, the problem is solved directly, i.e., no reference is made to the dual. Second, there is no pivoting process in its customary form.
Thus, the elemental transformations are applied until the problem is ready for the bound escalation method, and then the machinery for the latter is set into action. Those who wish may relate these two steps to a form of deferred pivoting and abbreviated pivoting, respectively, but attempts to salvage the pivoting concept are inessential to understanding the process. Third, no use is made of slack variables to transform inequations into equations. Fourth, the method works generally to

satisfy some set of constraints simultaneously with the bound escalation method rather than taking the more narrow immediate view of satisfying a single constraint. Fifth, the bound escalation method leaves all constraints of the bounding form satisfied, whereas the pivoting process of other integer algorithms may not in one step completely satisfy the constraint to which they are applied. Sixth, because there is no customary pivot operation, a choice among eligible pivotal constraints is replaced by a choice of another sort, i.e., that of the sequence of elemental transformations with which to establish a bounding form.

We will now lay down the basis of the algorithm. Proofs of the lemmas to follow will be found in Appendix I.

I. The Elemental Transformations.

Consider the set of transforms $T = \{ T_1^{rs}, T_2^{rs} \}$, $r, s = 1, 2, \ldots, m$, $r \neq s$, where we define the components t_{ij} of the m×m matrix T_k^{rs} by

$$t_{ij} = \begin{cases} (-1)^k & \text{if } i = r,\ j = s \\ \delta_{ij} & \text{(the Kronecker delta) otherwise.} \end{cases}$$

For example, if $m = 4$ we may write

$$T_1^{24} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

It is immediately seen that the matrix

$$D = T_k^{rs} A$$

is the same as A except in row r in which case the components of D are given by

$$d_{rj} = a_{rj} + (-1)^k a_{sj} \qquad j = 1, \ldots, m + n$$

The elements of T are called elemental transformations, and consist simply of row additions and subtractions in the A matrix, as can be seen by the foregoing remarks.

We now introduce the following two problems

(3) Minimize $wb + b_0$

subject to $wA \geq c$

and

(4) Minimize $z(Rb) + b_0$

subject to $z(RA) \geq c$,

which we relate by the following two lemmas.

Lemma 1. Assume that (3) and (4) have finite optima, and that R may be decomposed into a product of elemental transformations. Then the vector w^* is a feasible (optimal) integer solution of (3) if and only if $z^* = w^*R^{-1}$ is a feasible (optimal) integer solution of (4).

Lemma 2. Let R be any transformation for which Lemma 1 is universally true for the standard integer programming problem of formulation (1) (i.e., for any A^o, b, and c^o satisfying the finite optima restriction), and let R_0 be obtained from R by reindexing two columns of R. Then either R or R_0 can be expressed as a product of elemental transformations.

Since reindexing two columns of R corresponds to reindexing two rows of the augmented matrix (b A), and since the latter merely changes the order in which we list the variables without changing the basic problem in (3), Lemma 2 shows that there is a form of universality inherent in the elemental transformations in their application to the integer programming problem.

For the present algorithm we wish to restrict R somewhat more than in Lemmas 1 and 2. To facilitate the ensuing discussion we record problems (3) and (4), ~~inxthe~~ respectively, in the tabular forms

(3')

$-b$	A
b_0	c

(4')

$-Rb$	RA
b_0	c

and define the augmented matrices corresponding to the upper portions of (3') and (4') to be the <u>tableau matrices</u> of problems (3) and (4). We observe that the first column of the table (3') corresponds to the objective function $wb + b_0$, and that the successive columns identify the constraints given by the matrix inequation $wA \geq c$.

One of the restrictions we wish to put on R is that the tableau matrix of (4) is lexicographically negative (by row) whenever the tableau matrix of (3) is lexicographically negative. The other is that the necessity for any feasible solution of (3) to be nonnegative implies that any feasible solution of (4) will be also.

If we factor R into elemental transformations, we see that we are assured of both of these provided each successive factor of R assures them. For after each step we may redefine A and b to equal the matrix and vector RA and Rb just obtained, and reapply the result. By means of this reasoning we take care of the nonnegativity restriction in <u>Lemma 3</u>. Assume that any feasible solution of (3) must be nonnegative, and let z^* denote any feasible solution of (4). Then $z^* \geq 0$ is implied by either of the following,

(i) $R = T_1^{rs}$ for any r and s $(r \neq s)$.

(ii) $R = T_2^{rs}$ and there exists a j such that

$c_j \geq 0$, $a_{ij} \leq 0$ for $i \neq s$, and $-a_{rj} \geq a_{sj} > 0$.

Lemma 3 says that we may always assure that $z^* \geq 0$ by subtracting one row from another in the tableau matrix. If we add one row to another we may still be assured that $z^* \geq 0$ provided we identify a j such that (a) $c_j \geq 0$, and (b) the jth column of A will have all components nonpositive after the addition except a_{sj}.

This special form of the jth column of A in conjunction with $c_j > 0$ is of particular interest to us. It provides the fundamental unit of the framework on which we operate with the algorithm to converge to an optimal solution. Drawing on Lemma 3 we now show how we may manufacture this type of column by a sequence of elemental transforms, simultaneously preserving the desired restrictions on the problem form.

Lemma 4. Assume that (3) ~~is~~ has finite optima, satisfies the nonnegativity and lexicographic ordering restrictions, and that the component c_J of c is positive. Further assume that the tableau matrix consists entirely of integers. Then the following method defines an R in a finite number of steps so that problem (4) satisfies the same restrictions and so that the Jth column of RA contains exactly one positive component.

1. Begin with $R = I$.

2. Select a positive component of $(a_{\cdot J})$, where ~~is~~ $(a_{\cdot J})$ denotes the Jth column of A

If there are no other positive elements in $(a_{\cdot J})$ the procedure is completed. Otherwise pick a second positive component of $(a_{\cdot J})$ and define the subscripts r and s so that a_{rJ} is the component associated with the lexicographically smaller row of the tableau matrix and a_{sJ} is the component associated with the larger row.

4. Redefine A and b to be $T_1^{rs} A$ and $T_1^{rs} b$ (i.e., subtract row s from row r in the tableau matrix), redefine R to be $T_1^{rs} R$, and return to instruction 2.

It is evident that there may be a variety of ways for reducing the column $(a_{\cdot j})$ to the desired form by the method of lemma 4, some of which may be more efficient in terms of the number of steps required than others. One immediate way that would generally reduce the number of steps would be to replace T_1^{rs} in instruction 4 by $(T_1^{rs})^h$, where h is the largest integer multiple of row s which when subtracted from row r will leave the resulting row vector lexicographically negative. However, as we shall see, the different ways of reducing $(a_{\cdot j})$ also may be more or less efficient in terms of the extent to which we can exploit the structure of the resulting tableau matrix. Hence at this point we choose not to be restrictive.

II. The Bounding Form and the Bound Escalation Method.

Let D be a matrix whose columns correspond to some subset of the columns of A, and let d be the row vector which corresponds to c in exactly the same way that ~~Bxxx~~ D corresponds to A. Further suppose (i) each column of D contains exactly one positive component, and (ii) at least one of the entries of d is positive. Finally, let B be the matrix obtained from D by eliminating all rows in which no positive element appears. Then we define B to be a bounding form of A.

In the extreme, D may consist of a single column of A and d a single positive component of the c vector, which is the configuration which Lemma 4 shows how to manufacture. In this case the matrix B consists of the single positive component of D. We observe that the inequation

$wD \geq d$ consists simply of a subset of the constraints defined by $wA \geq c$. The bound escalation procedure is based on the principle that we may shrink w to x in the same way that D was reduced to B, and replace $wD \geq d$ by $xB \geq d$, so that whenever we find an x^* to satisfy the latter, we have implicitly a w^* which satisfies the former. Moreover, the form of x B enables us to find an x^*, hence a w^*, such that the constraining relation $w \geq w^*$ must be satisfied by any feasible solution of (3). The following lemmas show how such a relation (which we will soon show how to exploit) may be developed.

<u>Lemma 5</u>. Let D, B, d, and x be given as above. We assume that (3) has finite optima with all variables constrained nonnegative. Then in a finite number of steps the following procedure will find constrained lower bound values for the components of w which will satisfy $wD \geq d$.

1. Let $x = 0$.

2. If all components of d are nonpositive, go to instruction 4. Otherwise, pick a positive component, say d_j. (For explicitness, a reasonable rule is to let d_j be the largest positive component.)

3. Increment x_k by $\min_k \{d_j/b_{kj}\}$, where b_{kj} is the unique positive element in the jth column of B. Redefine d to be $d - \{d_j/b_{kj}\}(b_{k\cdot})$, where $(b_{k\cdot})$ is the kth row of B, and return to 2.

4. The lower bounds for the components of w which correspond to components of x are given by the x vector, the remaining lower bounds being 0. (At least one of the components of x must be positive.)

(The Next Page Is Page 10.)

We will illustrate the method of lemma 5 with the following example problem, already in tabular form.

-9	15	-1	24
-4	3	0	-3
-1	-7	3	-15
20	-4	10	-8

From the last two columns we identify a bounding form which we set up in a smaller table to demonstrate the method.

		Δw_1	Δw_3
-1	24		
3	-15		
10	-8		
-2	52		4
1	-20	3	
-2	-5		1

The successive adjustments of the d vector are shown in the additional rows below the bounding form and the original d vector. Beside each vector is the increment of the variable which created that vector out of the previous one. Hence we end up with $w_1 = 3$ and $w_3 = 5$. It may be verified by substitution that these values satisfy the constraints associated with the bounding form, and in fact in this case satisfy all the constraints of the problem.

While the procedure just given is sufficient to find the desired lower bound values of w, the complete bound escalation method is designed to shortcut this procedure by exploiting certain properties of the bounding form. We turn to a consideration of these properties with the following definitions.

Let B and E be bounding forms of A such that B is a submatrix of E (or the same as E). Then we will call B a <u>subform</u> of E.

If each row of the bounding form E has exactly one positive element, we will call E a <u>prime bounding form</u>. Similarly, B will be called a <u>prime subform</u> of any bounding form E if it is a subform of E and a prime bounding form.

We note that every bounding form has at least one prime subform, and also that every subform of a prime bounding form is a prime subform.

In the following examples of matrices with their associated d vectors below them, (a) defines a bounding form, (b) defines a prime subform of (a), and (c), though it may be abstracted from (a), does not define a bounding form at all since none of the components of its d vector are positive. Finally, (d) fails to define a bounding form on two counts; the first column contains more than one positive component and the second row contains none. However, the form of the last two columns is such that we may permissibly create a bounding form out of them by removing the second row.

(a)
$$\begin{array}{rrrr} 6 & 3 & -1 & -3 \\ -2 & -1 & 2 & -1 \\ 0 & -5 & -1 & 3 \\ \hline 4 & 0 & 1 & -2 \end{array}$$

(b)
$$\begin{array}{rrr} 6 & -1 & -3 \\ -2 & 2 & -1 \\ 0 & -1 & 3 \\ \hline 4 & 1 & -2 \end{array}$$

(c)
$$\begin{array}{rr} 3 & -3 \\ -5 & 3 \\ \hline 0 & -2 \end{array}$$

(d)
$$\begin{array}{rrr} 2 & -1 & 3 \\ -2 & -1 & -2 \\ 3 & 7 & -4 \\ \hline 3 & 9 & -6 \end{array}$$

Lemma 6. Let E be a prime bounding form, and B a subform of E. Let d be associated with B as before, and define the vector $r = dB^{-1}$. Then there exists a unique subform B^* of E such that (i) B^{*-1} consists only of nonnegative components, (ii) each r_k^* is positive, (iii) if w^* is any feasible solution of (3), then $w_{j_k}^* \geq \langle r_k^* \rangle$, where the index j_k corresponds to k as the indices of c correspond to those of d^*, (iv) if any other subform B of E satisfies properties (i) and (iii), then B^* implies a value for each of the components of w^* (in the manner of (iii)) that is at least as large as implied by B.

Lemma 7. We use the notation of the preceding lemma, and let h be the vector associated with E as d is associated with B. Then the following method finds the values of the r_k^*.

We assume for convenience that E is indexed so that its positive elements lie along the principal diagonal.

1. Select any positive h_j in h for which the subscript

j has not been chosen previously. If none exists, go to instruction 4.

2. Locate the corresponding positive entry e_{jj} in the E matrix and reduce the E matrix and h vector by the Gaussian reduction method on the jth row of E. All elements of the jth row become 0 except the new e_{jj}, which is 1.

3. Redefine E and h to be the matrix and vector resulting from step 2, and return to 1.

4. The values of the r_k* are read directly from the final h vector. B* is identified as the subform of E whose columns correspond to the positive components of h, and r_k* is the kth such positive component. We obtain the corresponding lower bounds for w as in lemma 6.

We remark that in step 2 above if e_{jj} turns out to be nonpositive the problem lacks finite optima and the process may be terminated.

We may illustrate the foregoing method with the same problem used to illustrate the method of lemma 5. We write below only the bounding form, which we have indexed to correspond to the specifications of lemma 7.

$$
\begin{array}{cc|}
24 & -1 \\
-15 & 3 \\
\hline
-8 & 10 \\
 & \uparrow
\end{array}
$$

[illegible] column we choose the [illegible] component [illegible] row and obtain the following [illegible] 5.

19	-1/3
0	1
42	10/3

1	0
0	1
42/19	232/57

We obtain the integer lower bounds for w_1 and w_3 by rounding the given values upward, or $w_1 = 3$, $w_3 = 5$. In this case the bounds are the same as found by the method of lemma 5, which we know satisfy the constraints of the bounding form. This will not invariably happen, nor will the amount of computation required by the two methods usually be so nearly the same. Generally speaking, the bounds implied by the method of lemma 7 will fall somewhere below those implied by the method of lemma 5. On the other hand, there may be computational savings of several orders of magnitude by using the bounds of lemma 7 to provide the method of lemma 5 with a head start. The extent of the savings will depend both on the nature of the E matrix of lemma 7 and its associated h vector. In general, the method of lemma 7 should be bypassed only if h has not more than a single positive component h_j, and the ratio h_j/e_{jj} is less than or equal to 1.

With the following two definitions we complete the inventory of the tools of the bound escalation method, which is presented immediately following.

We will call B the <u>distinguished bounding form</u> of A if all other bounding forms of A are subforms of B. We will call the prime subform E of D a <u>maximal prime subform</u> if E has the same number of columns as D has rows.

The Bound Escalation Method:

1. Identify the distinguished bounding form B of A.

2. Select a maximal prime subform E of B and apply the procedure of lemma 7 to E. (The method of lemma 7 may be bypassed if h has only one positive component h_j, and $h_j/e_{jj} \leq 1$.)

3. Use the lower bounds obtained from step 2 as starting values for the components of x in lemma 5, define the starting value of d as $d = xB$, and apply the method of lemma 5 to the columns of B corresponding to E. If any constraints of B are left unsatisfied, apply lemma 5 to all of B.

III. <u>Translation of the Problem and Convergence to Optimality</u>.

We now show how to take advantage of the lower bounds produced by the bound escalation procedure in the last stage of the algorithm. Consider the problem

(5) Minimize $wb + t_0$, $t_0 = w^0 b$

$$\text{subject to } wA \geq c - w^0A.$$

We relate (5) to (3) by means of

<u>Lemma 8</u>. Let w^0 be any vector, and assume the (3) and (5) have definite optima. Then if $\hat{w}$ is a feasible (optimal) solution of (5), $w^* = \hat{w} + w^0$ is a feasible (optimal) solution of (3), and conversely.

Lemma 8 tells us that we may let w^0 be the vector of lower bounds for w established by the bound escalation procedure and replace problem (3) by problem (5). Since $w^* \geq w^0$, this will keep $\hat{w} = w^* - w^0$ nonnegative. We know that the new c vector (equal to the old $c - w^0A$) will be nonpositive for all constraints associated with the old bounding form. If any components of c are still positive we may use the method of lemma 3 to create a bounding form that is associated with at least one of these components, and repeat the process. Suppose that on some step of exchanging (3) for (5) it turns out that $c - w^0A$ becomes entirely nonpositive. Since with each application of lemma 3 we have maintained $b \geqq 0$, and also constrained w to be nonnegative, a trivial optimal solution to (5) is to let $\hat{w} = 0$.

Thus to assure that the algorithm works and that we can take advantage of it if there are two problems: we must be able to force c eventually to become nonpositive, and we must be able to salvage the w^* which gives the optimal solution to the original problem given $\hat{w} = 0$ in the final. After formally outlining the steps of the algorithm we will address ourselves to these two problems.

General Form of the Algorithm:

1. Determine whether all components of c are nonpositive. If so, we are through. Otherwise,

2. Create a bounding form by the method of lemma 4. Make any legitimate row additions as desired.

3. Apply the bound escalation method to find a lower bound vector w^0 which will satisfy all constraints of the bounding form.

4. Redefine b_0 to be $w^0b + b_0$, redefine c to be $c - w^0A$, and return to 1.

Table (3') provides the vehicle for the bookkeeping of the method, since all updating is handled by adding or subtracting rows in (3'), followed by subtracting positive integer linear combinations of the upper rows from the bottom row.

The following lemma shows that with the bookkeeping provided by table (3'), the problem of finding the optimal solution to the original problem given that $\hat{w} = 0$ in the final becomes trivial.

Lemma 9. The optimal solution w^* to the original problem is the negative of the vector in the final table in the location corresponding to the portion of the c vector that was originally the 0 vector associated with the constraint $wI \geqq 0$.

We complete the specification of how convergence to optimality may be guaranteed with

Lemma 10. Assume that (3) has finite optima, and let j be a subscript for which some $c_j > 0$. Then if A, b, and c are integral, any rule for generating bounding forms which specifies that the jth column of A is eventually included in a bounding form will assure that an optimal solution will be found in a finite number of steps. (The inclusion of c_j in a bounding form is of course unnecessary if c_j becomes non-positive.)

IV. Rules of Choice.

The freedom allowed for selection among alternatives in the algorithm is immense. We outline below the various provinces in which choice occurs.

C-1. In the selection of columns of A as candidates for translation into a bounding form.

C-2. In the selection of elemental transformations to create a bounding form.

C-3. In the choice of when to apply the bound escalation method.

C-4. In the choice of which maximal prime subform to use in the first stage of the bound escalation method when more than one is available.

To obtain a problem solution expediently and efficiently there are several considerations which suggest how the range of alternatives may be narrowed. We examine the four regions of choice in more detail below, introducing such considerations as we go.

C-1 is perhaps one of the easier choices. Certainly, in the selection of columns of A as candidates for translation into the bounding form, we must include one associated with a positive component of the c vector, and it is not unreasonable to choose a set of such columns which are already close to having the bounding form.

C-2 is critical. In lemma 3 it was shown that elemental transformations of the first type (row subtractions in the tableau matrix) would always suffice to keep the variables nonnegative. Hence elemental transformations of the second type (row additions), which require special circumstances before they are permissible, are not strictly necessary. They are, however, frequently desirable. The reason is as follows. We refer again to the formulation of lemma 1 in which the transform R is applied to problem (3) to yield (4). Consider the effect of the transform R on the new variables given by $z = wR^{-1}$ when R is equal to T_1^{rs} and T_2^{rs}. Since T_1^{rs} and T_2^{rs} are inverse to each other, when $R = T_1^{rs}$ z is the same as w in every component except $z_s = w_r + w_s$. When $R = T_2^{rs}$ we have $z_s = w_s - w_r$, and again the remaining components of z and w are the same. Hence in using transformations of the first kind we are generally raising the solution values of the variables, while in using those of the second kind we are generally lowering them. Insofar as raising the solution values may imply that more effort is needed to locate the constrained lower bound, row additions would seem to be preferable to row subtractions when they are available. This surely is the case when by a row addition it is simultaneously possible to add a new constraint to the bounding form.

But this does not exhaust the tally of considerations [illegible] by C-2. By the argument just indicated, it would appear that of two methods for adding a column to the bounding form the one that required fewer row subtractions would be preferable. On the other hand, this is highly [illegible], since some row subtractions will create new variables with much higher bounds than will be created by others. One approach to greater selectivity is to try to keep the objective function vector as large as possible for the rows that seem likely to find their way into a bounding form.

[illegible] escalation method is in fact the power [illegible] and in the main other considerations should be made subordinate to getting as many rows and columns of A into the bounding form as possible.

C-3 is perhaps an easier choice, for which experience may provide an answer. The question is simply that of whether to apply the bound escalation method as soon as a bounding form is created, or to wait and try to build a larger structure for the method. Intuition suggests that for easy problems the solution may well be found by a few simple steps before a complex bounding form is created. On the other hand, for harder problems it could well be that trying to advance toward the solution before a good bounding form is created would be a wasted effort in the sense that the escalation method will take the variables up to the same point in roughly the same amount of time in any case. Limited experience with problems small enough to be solved by hand seems to support this notion.

C-4 appears at this point not too critical. Whenever two or more columns of the distinguished bounding form have their positive components in the same row, one expedient rule for determining which one to use in the maximal subform is simply to select the column associated with the largest element in the c vector. This might reasonably be made subordinate to a rule which gives first preference to columns that are not located where the I matrix was originally, since we would generally expect constraints derived from the original nonnegativity restrictions to be weaker than the others. A more refined rule would be to apply the method of lemma 7 to all columns of the distinguished bounding form, rather than only to a prime subform. Complete reduction could be carried out using those columns whose positive components were unique to their rows, until none were left that could legimately be used. The delayed choice could then be made by picking from a set of eligible columns that have their positive components in the same row the column J so that, in the terminology of lemma 7, $h_J/e_{JJ} = \max(h_j/e_{jj})$, where j ranges over the columns indicated. (We require, of course, that the ratio be positive.) More refined rules may be invented if the need for them arises.

V. A Specific Algorithm and Example Problems.

In order to illustrate the workings of the algorithm we will arbitrarily settle on a few simple rules of choice. The specific method which results may be outlined as follows.

1. If the problem does not exhibit a bounding form, go immediately to instruction 2. Otherwise, identify the maximal subform E. If there is a choice to make among more than one column for inclusion in E, give first preference to those not initially in the I matrix, and of the remainder, select the one associated with the largest component of the c vector. Make any permissible row additions with respect to the columns of A associated with E. (If step 3 has previously been carried out, exclude any additions which would reverse a subtraction performed in step 3.) Apply the bound escalation procedure until all constraints of the bounding form are satisfied, and adjust the c vector.

2. If c is nonpositive, the problem is solved. Otherwise, of those columns j of the A matrix for which $c_j > 0$, select the column j which has the fewest number of positive components. If there are ties, restrict j to the tied columns and choose j so that $c_j = \max(c_j)$.

3. Consider only those rows of the tableau matrix for which $a_{ij} > 0$. Pick the lexicographically greatest row I from among them, and for each remaining row i restricted as above evaluate (a) the least (positive integer) multiple of row I which when subtracted from row i will make the resulting a_{ij} nonpositive, (b) the greatest multiple of row I which can be subtracted from row i and leave row i lexicographically negative in the tableau matrix. Pick from (a) and (b) the multiple that is smallest, carry out the indicated subtraction for all i as defined, and return to instruction 1.

Any ties not resolved by the method may be broken by selecting the alternative with the least index. We now solve the following example problems with the method as outlined. We have partitioned the tables to segregate the portion corresponding to the starting identity matrix in order to keep track of the solution values of the variables.

Problem 1. Minimize $10w_1 + 14w_2 + 21w_3$

subject to

$$\begin{aligned} 8w_1 + 11w_2 + 9w_3 &\geq 12 \\ 2w_1 + 2w_2 + 7w_3 &\geq 14 \\ 9w_1 + 6w_2 + 3w_3 &\geq 10 \\ w_1, w_2, w_3 &\geq 0 \end{aligned}$$

Column J defined in step 2 is indicated by the arrow.

-10	8	2	9	1	0	0
-14	11	2	6	0	1	0
-21	9	7	3	0	0	1
0	12	14 ↑	10	0	0	0

After the first application of step 3, we obtain the following table. The c vector has been adjusted by identifying the single element maximal subform defined in step 2 (the 9 in the first row and fourth column), and obtaining the lower bound for its associated variable.

-10	8	2	9	1	0	0
-4	3	0	-3	-1	1	0
-1	-7	3	-15	-2	0	1
20	-4	10	-8	-2	0	0

We see that the left half of this table is the example problem used to illustrate the two methods underlying the bound escalation procedure. Identifying the two by two maximal subform in the third and fourth columns, we already know that the lower bound solution for it is given by $w_1 = 3$ and $w_3 = 5$. Adjusting the c vector by subtracting 3 times the first row and 5 times the third row we obtain

52 | -11 -2 -5 | -1 0 -2.

Hence the final answer to the problem is $w_1 = 1$, $w_2 = 0$, $w_3 = 2$.

The preceding problem was taken from [3], where it is used by Gomory to exemplify his all integer algorithm.

(Using two different rules of choice, he solved it in 4 pivots and 3 pivots, respectively. See Appendix II for a more complete comparison of the two methods.) Using the simple rules of choice specified above, we obtain the same sequence of tables as Gomory does, allowing for representational differences, though we operate on them in different ways. We now present a problem for which Gomory's method [illegible] give the sequence of tables obtained by our method. The bounding forms associated with tables (2) and (3) are shown immediately following the respective tables.

Problem 2--Initial Table.

(0)

-6	-5	8	1	1	0	0
-4	-9	14	-3	0	1	0
-10	4	-6	3	0	0	1
0	0	0	1 ↑	0	0	0

(1)

-6	-5	8	1	1	0	0
-4	-9	14	-3	0	1	0
-4	9	-14	2	-1	0	1
0	0	0	1 ↑	0	0	0

(2)

-2	-14	22	-1	2	[illegible]	[illegible]
-8	0	[illegible]	-1	-1	1	1
-4	9	-14	2	-1	0	1
[illegible]	5 ↑	-8	0	-1	0	0

22 -1

-14 [illegible] Δw_1 Δw_3

[illegible]

[illegible] 1

[illegible] 1

(3)	−6	−5	8	1	1	0	0
	−8	0	0	−1	−1	1	1
	−1	9	−14	2	−1	0	1
	58	−1	0	−14	−3	0	−4

−5	8		
9	−11	Δw_3	Δw_1
5	−8		
−4	6	1	
1	−2		1
−8	12	1	
2	−4		2
−7	10	1	
3	−6		2
−6	8	1	
−1	0		1

The preceding example can be used to highlight an interesting alternate application of the algorithm. In the last step, where the method of Lemma 5 was applied to the 2X2 maximal subform, it would have been possible to proceed somewhat differently. The 2X2 subform together with its associated d vector define "collapsed" constraints in nonnegative variables which must be satisfied by any feasible solution to the complete problem. If we make the nonnegativity restrictions explicit we can write the subform with an adjoined identity matrix, as follows.

-5	8	1	0
9	-13	0	1
5	-8	0	0

Instead of using the bound escalation method without interruption, we will work with this expanded table and intervene whenever possible to make legitimate row additions. We then obtain the following sequence of tables, where at each step we have properly adjusted the d vector according to the previous table, and then made any legitimate additions.

-5	8	1	0
4	-6	1	1
-4	6	0	-1

-1	2	2	1
4	-6	1	1
1	-2	-1	-1

-1	2	2	1
1	0	7	4
-3	4	-2	-2

-1	2	2	1
1	0	7	4
-1	0	-6	-4

We see from the last table that the values for the variables which satisfy the constraints of the original 2X2 subform are 6 and 4, which of course are the same as obtained by applying the regular bound escalation method. The amount of computation for this case was perhaps slightly more than with the regular bound escalation method, but we note that had the problem not been completely solved the approach just used could be more advantageous (provided, as is the case here, that there do turn out to be legitimate additions possible).

The reason is that we are given the transform which will carry out the row additions in the complete table corresponding to those made in the collapsed table. As in the example problem, it is located in the portion of the final table that began as the identity matrix. Making use of this tends to speed up convergence, since, as we have seen, row additions generally tend to lower the solution values of the problem variables.

The two previous example problems were quite simple to solve both with Gomory's algorithm (which required 5 pivots for problem 2) and the algorithm presented in this paper. We have argued that for harder problems the potential efficiency of our algorithm is based on the bound escalation method, and on using the extensive freedom of choice available in order to develop "good" bounding forms. To give an indication of this potential, we will conclude our presentation of examples with a problem which Gomory's method finds somewhat more difficult, requiring between 47 and 150 pivots to solve, depending on the rule of choice. We have indicated [illegible] the bound escalation method follows in Table (1).

Problem 3.

(0)	[illegible]	[illegible]	[illegible]	1	1	0	0
	[illegible]	6	7	-7	0	1	0
	[illegible]	7	[illegible]	6	0	0	1
	[illegible]	[illegible]	[illegible]	0	0	0	0

(1)	-9	9	-6	-1	1	2	9
	-4	-8	7	-1	0	1	0
	-12	-1	-2	5	0	1	1
	549	0	-2	-2	-29	-102	-14

(1,0)	9	-6	-1	(1,1)	1	0	0
	-8	7	-1		-8/9	5/3	-17/9
	-1	-2	5		-1/9	-8/3	44/9
	7	6	9		7/9	32/3	88/9

(1,2)	1	0	0	(1,3)	1	0	0
	0	1	0		0	1	0
	-33/15	-8/5	38/15		0	0	1
	97/15	32/5	328/15		337/14	176/7	82/7

Rounding upward, starting values for the next stage are (25 26 12).

9	-6	-1			
-8	7	-1			
-1	-2	5	Δw_1	Δw_2	Δw_3
2	-2	0			
-7	4	1	1		
1	-3	2		1	
2	-1	-3			1
-7	5	-2	1		
1	-2	-1		1	
-8	4	0	1		
0	-3	1		1	
1	-2	-1			1
-8	5	-1	1		
0	-2	-1		1	

Hence the final values for these variables are (29 30 14), which gives the adjusted c vector in table (1).

Because of the space devoted to their representation, it may seem at first glance that the computations required with the second stage of the bound escalation method are excessive. However, reflection will show that a number of steps of this second stage may be carried out in the amount of time normally required for a single pivot.

The preceding problems are not to suggest in any dogmatic fashion how the performance of our method and Gomory's algorithm are likely to compare. As Appendix II argues, there are reasons to believe that Gomory's algorithm may possibly be more efficient for simple problems which both methods can handle relatively easily. However, some problems which are difficult to handle with Gomory's algorithm should not be too formidable with the present code, as the last example shows. G. L. Thompson has found a three variable three inequality problem for which Gomory's algorithm requires over 1500 pivots. Our algorithm creates a 3X3 maximal subform from the problem in four steps, and the bound escalation procedure applied to this subform is then able to manufacture the solution in the same way that it handled the third example problem above. In a related vein, the author has made a study of Gomory's all integer algorithm and developed a supplementary technique based on concepts similar to those underlying the algorithm of this paper which significantly reduced the number of pivots ordinarily required to obtain an optimal solution (see [2]). This technique may also be coupled with the present algorithm as a strategy for accelerating convergence.

We have, in the foregoing examples, used a very limited rule of choice in order to keep the exposition simple. In view of lemma 2, there is a sense of universality about the elemental transformations in their application to this problem, so that a major avenue toward gaining control over the integer programming problem may lie in attempts to learn the best ways of manipulating these transformations. Theorems about the existence of bounding forms, or of bounding forms with certain value for obtaining a problem solution, and the sequence of transformations designed to locate them, would certainly be desirable.

An alternate opportunity for manipulating the problem structure to obtain a good bounding form may be to create derivative inequations by adding together positive multiples of the current constraints. For example, the principle of Gaussian reduction in lemma 7 could be applied in the absence of a bounding form so long as no subtractions or divisions by negative numbers were permitted. Thus it might be possible to create a set of constraints (or even a single constraint) considerably nearer the bounding form than any in the problem table, and to use these derivative constraints as well as the original ones as a guide for the additions and subtractions of rows, and the subsequent alterations of the c vector. The derivative constraints would not have to be integral, since they would necessarily consist of rational numbers, and the proofs assuming integrality would remain valid. It is conceivable that a technique which combined these considerations with a good rule for selecting the elemental transformations might prove quite effective for certain classes of problems.

APPENDIX I

<u>Proof of Lemma 1</u>. R^{-1} exists since for each r and s $(r \neq s)$, Γ_1^{rs} and Γ_2^{rs} are inverse to each other. Since each elemental transformation consists entirely of integers, this also implies that both R and R^{-1} are integral, hence the relationship $z^* = w^*R^{-1}$ gives that either of z^* or w^* must be integer when the other one is. The direct correspondence of feasible solutions is observed immediately by substituting w^*R^{-1} in (4) under the assumption that w^* is feasible in (3), and by substituting z^*R in (3) under the assumption that z^* is feasible in (4). Finally, if w^* is optimal for (3), then z^* must be optimal for (4), or else there exists a $\hat{z}$ for which $\hat{z}(Rb) + b_0 < z^*(Rb) + b_0$, and hence $\hat{w}b + b_0 < w^*b + b_0$, where $\hat{w} = \hat{z}R$. But by the foregoing remarks, $\hat{w}$ is also a feasible solution of (3), which is a contradiction. The converse proceeds similarly.

<u>Proof of Lemma 2</u>. We must handle lemma 2 in several parts.

<u>Remark 1</u>. If R satisfies lemma 1 for the integer programming problem of formulation (1), for all A^0, b, and c^0 satisfying the finite optimality restrictions, then R must be integral.

<u>Proof</u>: From formulation (1) we are given that $A = (A^0 \quad I)$ and $c = (c^0 \quad 0)$, so that we may rewrite (3) and (4) respectively as

(3a) Minimize $wb + b_0$

$$\text{subject to } wA^0 \geq c^0 \text{ and } w \geq 0.$$

(4a) Minimize $z(Rb) + b_0$

$$\text{subject to } z(RA^0) \geq c^0 \text{ and } zR \geq 0.$$

Suppose that $b_i > 0$ for each component of b and $A^o = R^{-1}$.
Then (3a) and (4a) become

(3b) Minimize $wb + b_0$

subject to $wR^{-1} \geq c^o$ and $w \geq 0$,

(4b) Minimize $z(Rb) + b_0$

subject to $z \geq c^o$ and $zR \geq 0$.

We will show that for the proper choice of c^o problem (4b) must have finite optima when each element of b is positive.

Consider $c^o = \hat{c}$ in (3a), where $\hat{c}$ is integral and has all positive components, and let $A^o = I$. Then (3a) and (4a) reduce to

(3c) Minimize $wb + b_0$

subject to $w \geq \hat{c}$ and $w \geq 0$,

(4c) Minimize $z(Rb) + b_0$

subject to $zR \geq \hat{c}$ and $zR \geq 0$.

We see that (3c) is trivially and uniquely optimized for $b_i > 0$ by letting each component of w equal the corresponding component of $\hat{c}$. But then (4c) must have the finite optimum given by $z = \hat{c} R^{-1}$. From this we may conclude that any choice of c^o which leaves the solution set of (4b) nonempty also implies that (4b) has finite optima. Evidently (4b) must have finite optima whenever this is implied by the conjunction of $zR \geq 0$ and the objective function. But (4c) has the same objective function and the existence of finite optima there is implied by $zR \geq \hat{c}$ and $zR \geq 0$. Since $\hat{c} > 0$, $zR \geq \hat{c}$ implies $zR \geq 0$, and if the latter implies the absence of finite optima the former must also.

But since (4c) does have finite optima, we conclude by contradiction that (4b) does too.

We now show how we may obtain a further contradiction by assuming R is nonintegral and satisfies lemma 1. If R satisfies lemma 1, then the optimal solution $c^0 = \hat{c}R^{-1}$ of (4c) must be integral given that the optimal solution c of (3c) is integral. We observe moreover that $z = c^0 = \hat{c}R^{-1}$ is a feasible solution for (4b), since substituting this value in the two constraints gives $z = c^0 \geqslant c^0$, and $zR = c^0 R = \hat{c} \gtrsim 0$, where the inequations are more restricted than we have shown them, but in any case satisfied. Now suppose that R is fractional in some component in the ith row (r_{i_o}). We select $\hat{c}$ large enough so that $\hat{c} + (r_{i_o}) > 0$, and define $z^* = c^0 + (0\ 0 \ldots 1_i \ldots 0)$, where as before $c^0 = \hat{c}R^{-1}$. Then z^* is a feasible solution of (4b) since clearly $z^* \gtrsim c^0$, and $z^*R = \hat{c} + (r_{i_o}) > 0$ satisfies the second constraint. We note from this last that whenever c is integral, z^*R is not. But the feasible solution of (3b) corresponding to z^* of (4b) is given by $w^* = z^*R$, which contradicts the assumption that R satisfies lemma 1. Therefore, R must be integral. (We note that if $c^0 = \hat{c} + (0 \ldots 1_i \ldots 0)$ then z^* is also an optimal solution of (4b) for which the corresponding optimal solution of (3b) is nonintegral.)

<u>Remark 2</u>. R satisfies lemma 1 if and only the determinant of R, $|R| = \pm 1$, given that R must be integer.

<u>Proof</u>. If $|R| = \pm 1$, then R is nonsingular and has an inverse, hence given that R is integer lemma 1 must be satisfied. On the other hand, if R makes lemma 1 true, R^{-1} must exist,

and by a reapplication of the reasoning of Remark 1 we know that it must be integral. But $|R| \; |R^{-1}| = |I| = 1.$ Since the determinants of integer matrices must be integers, we have $|R| = |R^{-1}| = \pm 1.$

Remark 3. We need only consider the case where $|R| = 1,$ since by reindexing two rows or columns as the lemma permits us to do, we may change the sign of the determinant if it is negative.

Remark 4. We will denote the transpose of a matrix by the prime (') superscript. Let h be a column vector of R (or any integral column vector). Then there exists a matrix X which can be expressed as a product of elemental transformations such that $h'X' = (0 \; 0 \; \ldots \; 0 \; k)$ where k is positive.

Proof: We need only to show that the remark holds for $h' = (h_1, h_2)$. If h_1 and h_2 are both positive or both negative we may proceed by always subtracting the smaller (in absolute value) from the larger, giving a strictly monotone decreasing sequence of absolute values until, (since they are integer) one of the components is zero. If we are left with $(g \;\; 0)$, where we do not specify whether g is positive or negative, we may change it to $(0 \;\; g)$ by the sequence $(g \;\; 0)$, $(g \;\; g)$, $(0 \;\; g)$, using the obvious additions and subtractions. If we end up with $(0 \;\; g) = (0 \;\; -k)$, we may obtain $(0 \;\; k)$ by the sequence $(0 \;\; -k)$, $(-k \;\; -k)$, $(-k \;\; 0)$, $(-k \;\; k)$, $(0 \;\; k)$. If h_1 and h_2 are of different signs we may first add the larger in absolute value to the smaller and proceed as before.

Remark 5. Using the method of Remark 4, we may create an

X which is the product of elemental transformations which will transform R into the identity matrix. Lemma 2 follows at once.

Proof. Select first the last column of R, and reduce all components to 0 except the bottom one which we leave positive. Then move to the next to the last column of R, and exclude the bottom row, accomplishing the same result with 0's in all rows except the next to last and possibly the last. Since these row additions and subtractions do not involve the last row, none of the 0's in the last column will be changed. Repeating this process, eventually all elements to the right of the main diagonal will be zero, and all elements along the main diagonal positive except possibly the one in the first row. We know that the determinant of this final matrix R is given by $|X R| = |X||R| = 1 \cdot 1 = 1$, since the determinant of each elemental transformation is 1. Finding the value of $|XR|$ by cofactors of the last column, we see that the bottom element on the main diagonal, which is positive, must be 1, and its minor must also be 1. Proceeding successively from minor to minor we apply the same argument to see that all the diagonal elements must be 1, including finally the last. Thus we may readily make all remaining nondiagonal entries in the matrix zero by adding or subtracting the appropriate integer multiple of the diagonal elements. Hence we have found a product of elemental transformations X such that $X R = I$. Letting $S = X^{-1}$, the lemma is proved.

<u>Proof of Lemma 3</u>. Since Γ_1^{rs} and Γ_2^{rs} are inverse to each other, when $R = T_1^{rs}$ we have $z^* = w^* T_2^{rs}$, and z^* is the same as w^* in every component except $z_s^* = w_r^* + w_s^*$. Hence z^* is nonnegative whenever w^* is. When $R = T_2^{rs}$, $z^* = w^* \Gamma_1^{rs}$, and z^* and w^* are the same except for $z_s^* = w_s^* - w_r^*$. Hence we must assure that z_s^* is nonnegative in some other way. In the constraint $z(RA) \geq c$, we have the Jth column of RA the same as the Jth column of A except in the <u>rth</u> row, in which we find the new coefficient $(a_{rj} + a_{sj})$. Under condition (ii) of lemma 3 this coefficient must be nonpositive, as must all other coefficients in the Jth column except a_{sj}. We may rewrite the constraint associated with column J as

$a_{sJ} z_s \geq c_J + L(z_k; k \neq s)$, where L is a nonnegative linear combination of the z_k for k other than s. Since for any feasible solution w^* of (3) we have $z_k^* = w_k^*$ for $k \neq s$, and since $c_J \geq 0$ by (ii), z_s^* must be nonnegative in order to satisfy the above constraint.

<u>Proof of Lemma 4</u>. As long as we have two positive integral elements in the Jth column we may always subtract the one in the lexicographically larger row from the one in the lexicographically smaller row while maintaining both lexicographic ordering, and by lemma 3, nonnegativity. Since at least one of the positive coefficients in the Jth column is reduced by an integer amount at each step, as long as more than one exists, eventually all but one must become nonpositive.

Proof of Lemma 5. Because of the form of D, each individual constraint from the constraint set $wD \geq d$ may be written in the form $d_{ij}w_i \geq d_j + L(w_k; k \neq i)$ where d_{ij} is the unique positive component in the Jth column of D, and L is a nonnegative linear combination of the w_k for $k \neq i$. Since B contains exactly those rows of D in which the positive components appear, we may rewrite the above as $b_{ij}x_i \geq d_j + L_0(x_k; k \neq i) + L_1(w_h; w_h \neq x_k)$ where again b_{ij} is the unique positive component in the jth column of B, L_0 is a nonnegative linear combination of the remaining x_k, and L_1 is a nonnegative linear combination of those w_h which are not represented by any x_k. Since we require any feasible solution to be in integers, we may write

$$x_i \geq \langle\langle (d_j + L_0(x_k; k \neq i)/b_{ij} \rangle\rangle.$$

We know that each of the x_k must be nonnegative, hence if there is any d_j that is positive we obtain a positive lower bound for the corresponding x_i, letting $L_0 = 0$. But as soon as this lower bound for some x_i is known, then we may compute a lower bound for the L_0 associated with each of the other x_i, hence giving new bounds for these x_i, as in step 3 of the method of lemma 5. If we wish to register only the incremental values given to the x_i in each step we may redefine each d_j to equal the old $d_j + L_0$. As long as any of these adjusted d_j remain positive we may increment the lower bounds of the corresponding x_i. Assuming the feasible solution set contains finite points, the process must eventually stop since we are incrementing the variables by integer amounts. At this point the constraints of the original inequation $wD \geq d$ must all be satisfied, for otherwise one of the adjusted d_j would still be positive.

Proofs of Lemma 6 and Lemma 7. We combine the proofs of these two lemmas since the justification of the method of lemma 7 proves both. We assume that we have adjoined the identity matrix above E, so that when we have completed the Gaussian reduction we will be able to identify the inverse matrix B^{-1} of the submatrix B of E which has been changed to an identity matrix. As stated in lemma 7 we assume that E is indexed so that its positive elements lie along the main diagonal. Suppose $h_j > 0$ and we carry out Gaussian reduction by the method of lemma 7 with the jth column of E. We will denote the values of the coefficients after the reduction by the prime (') superscript. Then the formula for the reduction as it affects E and the adjoined h vector is

$$e'_{ik} = \begin{cases} e_{ik}/e_{jk} & \text{for } k = j \qquad (1) \\ e_{ik} - e_{jk} \cdot e_{ij}/e_{jj} & \text{for } k \neq j \qquad (2) \end{cases}$$

where i ranges over all the rows of E. Similarly,

$$h'_k = \begin{cases} h_k/e_{jk} & \text{for } k = j \qquad (3) \\ h_k - e_{jk} \cdot h_j/e_{jj} & \text{for } k \neq j \qquad (4) \end{cases}$$

From the form of E, $e_{ik} > 0$ if and only if $i = k$. We will show that this relation continues to hold after the reduction with the possible exception that e'_{kk} becomes nonpositive when $h'_k < 0$. Since e_{jj} is positive, obviously the abovementioned relation is not changed by (1). In (2) we observe that e_{jk} is always nonpositive, and e_{ij} is nonpositive except when $i = j$. Thus $e_{jk} \cdot e_{ij}/e_{jj}$ is nonnegative for $i \neq j$, and subtracting it from e_{ik} leaves $e'_{ik} \leq e_{ik}$ for $i \neq j$. If $i = j$, then $e'_{jk} = e_{jk} - e_{jk} \cdot e_{jj}/e_{jj} = 0$. Thus the reduction insures that none of the e_{ik} which were originally nonpositive will later become positive. Since by this method we are always dividing through some constraint by a positive element (e_{jj}), followed by adding a nonnegative multiple ($-e_{jk}$) of the resulting constraint to each of the others, we are preserving the direction of the inequalities of the constraints of each step. Moreover, for the same two reasons, the inverse of the original matrix which is being implicitly calculated in this fashion must have all nonnegative components, since the identity matrix begins nonnegative. Finally, because the inequalities are preserved, e'_{kk} cannot become nonpositive for $h'_k > 0$ or else the fact that the variables must be nonnegative implies that the feasible solution set of the problem is empty. Thus the method of lemma 7 is well-defined.

The form of the constraints implies as in the proofs of the preceding two lemmas that the variable y_k associated with the kth row of E is bounded from below by the relation $y_k \geq \langle h_k/e_{kk} \rangle$. The corresponding value for y_j (with j identified as above) will be unchanged by the reduction step since $h'_j/e'_{jj} = h_j/e_{jj}$. We observe by the following that the lower bound for y_k, $k \neq j$ must either increase by the reduction step or stay the same (provided

$e'_{kk} \neq 0$). Since e_{jk} is nonpositive for $k \neq j$, and since we carry out the reduction step only when $h_j > 0$, we see from (1) that $h'_k \geq h_k$ for $k \neq j$. As pointed out earlier, $e'_{kk} \leq e_{kk}$ for $k \neq j$, hence $h'_k/e'_{kk} \geq h_k/e_{kk}$, as claimed.

When the process is completed we will have carried out the reduction step with every column for which h'_k is positive, so that, $e'_{kk} = 1$ for these columns, and the bound for y_k is simply given by $y_k \geq \langle h'_k \rangle$.

Let us denote the vector consisting of these positive h'_k by d', and let d denote the original vector corresponding to h as d' corresponds to h'. By the nature of the Gaussian reduction method, d' is the solution of the equation $xB = d$, where B is the submatrix of E which is transformed into the identity matrix in the process of changing d to d', and x corresponds to B as y corresponds to E. Thus we have $d' = d B^{-1}$, and we see that d' is the same as the r vector defined in lemma 6. We have already shown that B^{-1} consists of nonnegative components. We must finally show that no reduction which creates an identity matrix out of a different submatrix of E will imply a higher value for any y_k.

First, we observe that $x = d'$ satisfies the constraints $xB \geq d$, provided we do not require x to be integral, since in fact $d'B = d$. As a more general property of the Gaussian reduction method, d' is also a solution of $x\hat{B} = \hat{h}$, where $\hat{B}$ consists of the rows of E which appear in B, and $\hat{h}$ is equal to h in the components which correspond to d, and equal to [illegible]

[illegible]

we have $\hat{h} \geq h$, and hence d^* is a feasible solution of [illegible] $\geq h$. From this we note that [illegible] bounds given by [illegible] $\geq h_k$ where the h_k are in d^1, and given by y_r [illegible] constraint satisfy the constraint set [illegible] $\geq$ [illegible] that to prove that no greater bounds are [illegible] of the y_r by some other submatrix B of [illegible].

[illegible] we show that if we use any column for reduction whose components have all become nonnegative, then [illegible] can no longer be nonnegative, provided it exists. Since the reduction process will preserve the direction of the inequalities up to the point at which the first column with all nonnegative components is used for reduction, we know that if the h_k value currently associated with this column is negative, then the feasible solution set of the constraints is empty [illegible] there will always exist a positive original h vector for which the feasible space is empty. [illegible] readily [illegible] starting with any positive h vector, and [illegible] reaching the point where the specified column becomes [illegible] positive, add a large enough amount to the current h_k to make it positive (if it is not already). Adding the same amount to the original h_k will obviously produce the same result, since column k has not yet been used to reduce [illegible] the others.

As before, let us [illegible] to denote that [illegible] of the [illegible] vector which corresponds to the submatrix [illegible] [illegible] process [illegible]

[illegible]

in nonnegative variables. Then if d^{0} is [illegible] final vector [illegible] changing B to I, we have [illegible] that d' is the [illegible] to $x\,B = d$. Thus d' cannot be nonnegative, or else it would also give a feasible solution to $x\,B \geq d$, which we know does not exist. But the relation $d\,B^{-1} = d^{0}$ is also true. Thus, given that d is positive and that d' has a negative component, B^{-1} must contain a negative component as well.

For the last step we rule out using any column for reduction all of whose components in L have become nonpositive. Then, Gaussian reduction preserves the direction of the inequalities, and any values for d^{0} which are obtained by creating an identity matrix out of B must give legitimate lower bounds for the appropriate components of y. If any of these values are greater than those found by the method of lemma 7, we have a contradiction, since the latter values do in fact satisfy all the constraints. While some other submatrix B of L may imply the same lower bounds as [illegible] B^* identified by the method of lemma 7, it cannot do so if all components of d^{0} are positive without this being a self-contradiction, or unless it is the same as B^* after all. Thus B^* is unique, and the proof is complete.

<u>Proof of Lemma 8.</u> The pattern of this proof follows [illegible] the pattern of substitution [illegible] to show the correspondence of [illegible] solutions, as is used in proving lemma 1.

<u>Proof of Lemma 9.</u> So that we may use the notation of lemma [illegible] the optimal solution of the problem given in [illegible] From the remarks given in the paper,

$z^* = 0$. Thus if R is the transformation which changed the tableau matrix of the original problem into that of the final, we have by lemma 1 that the optimal solution of the problem given by the original tableau matrix and the final c vector is $w^* = z^*R$. In this case, since $z^* = 0$ $w^* = 0$ also. We now seek a vector w^0 so that the final c vector c_F and the original c vector c_I may be related by $c_F = c_I - w^0A$, and the final b_0 equal to the original b_0 plus w^0b. Then by lemma 8, given that $\hat{w}$ is the optimal solution to the problem

Minimize $wb + w^0b + b_0$

subject to $wA \geqq c_I - w^0A$

we are assured that $w + w^0$ is the optimal solution to the original problem

Minimize $wb + b_0$

subject to $wA \geqq c_I$.

Since $\hat{w} = 0$, the optimal solution for the original problem is simply w^0, provided that such a w^0 exists. We now show that this is the case.

In the successive steps of altering the c vector we began with c_I and applied lemma 8 at various points to obtain new values for its components. Let us denote the first c vector c_I by C_1, the second by C_2, and so forth. Correspondingly, let us denote the first w^0 used to change C_1 to C_2 by W_1, the second by W_2, etc. Then we have

$$C_2 = C_1 - W_1A_1$$
$$C_3 = C_2 - W_2A_2$$
$$\vdots$$
$$C_{i+1} = C_i - W_iA_i$$

where A_i denotes for each i the A matrix of the problem at the point when C_i was changed into C_{i+1}. From the foregoing $C_3 = C_2 - W_2A_2 = C_1 - W_1A_1 - W_2A_2$, and in general $C_{i+1} = C_1 - W_1A_1 - W_2A_2 - \ldots - W_iA_i$. But each A_i was obtained from the original A by some transformation R_i, so that by substituting $A_i = R_iA$, we obtain

$$C_{i+1} = C_1 - (W_1R_1 + W_2R_2 + \ldots + W_iR_i)\ A.$$

Hence letting C_{i+1} equal c_β we see that w^0 exists and is given by $w^0 = W_1R_1 + W_2R_2 + \ldots + W_iR_i$. (The identical sequence of reasoning using b in place of A and b_0 in place of c shows that the w^0 which translates the original b_0 into the final $b_0 + w^0b$ is the same as the one just obtained.) Rather than try to compute w^0 by the preceeding expression, however, we may find it more readily by restricting c_β and c_I to those components of c associated with the original constraints $wI \geq 0$. We denote this restricted c_β by $\hat{c}_\beta$, and obtain $\hat{c}_\beta = 0 - w^0I$, since the restricted c_I is equal to 0. Hence the optimal solution to the original problem is identified as $- \hat{c}_\beta$. This proves the lemma.

Proof of Lemma 10. The proof of this lemma is given by Ralph Gomory to demonstrate the convergence of his all integer algorithm (see [3]). We will not reproduce the proof here since the translations of terminology (interchanging rows and columns, replacing lexicographic positivity by negativity, substituting bounding form for pivot row) are straightforward, from which the applicability of the proof as it appears in [3] is immediate.

APPENDIX II

Gomory's All Integer Algorithm: Comparison and Contrast

Although Gomory develops his all integer algorithm in terms of dual variables and pivoting operations, it may be explained very simply by means of the concepts presented in this paper. Such an explanation will be useful for a clearer understanding of the relation of the two algorithms. As will be shown, Gomory's algorithm may be regarded essentially as a variant of the present algorithm in which (i) freedom of choice in applying the elemental transformations is exchanged for a restricted routine to insure methodical progression toward a solution with each step, (ii) after applying the restricted rule, a multiple of one of the constraining inequations is added to another to produce the type of constraint manufactured by lemma 4 in which exactly one coefficient is positive, (iii) all row additions possible in the new constraint are carried out, and (iv) the lower bound implied for the single variable with a positive coefficient in this new constraint is calculated, and the c vector adjusted accordingly.

We have indicated for the first example problem in section V that the restricted version of our method obtains the same sequence of tables as Gomory's method. This is not entirely accidental. The simple rule for row subtractions which we selected for our example method is the same one which Gomory's method uses in the first stage of his pivot operation. However, because the two methods operate somewhat differently on a given table, the actual row subtractions available to each may not coincide after the first few steps. Moreover, the rules for row additions with the two methods are slightly different.

From the standpoint of our algorithm these differences are primarily differences by design, while for Gomory's method they are mainly differences of necessity. One reason for this, as we have observed, is the limitation Gomory's algorithm imposes on the selection of elemental transformations in the first stage of the pivoting process. Another reason is an equally strong limitation imposed on the second stage of the pivoting process, in which Gomory's method uses the strategy of combining two of the problem constraints into a new one in order to find a lower bound for one of the problem variables. (It should be emphasized, in this regard, that Gomory's algorithm is not actually divided into the "stages" we have identified, nor is it designed to employ a "strategy" of combining two problem constraints to find lower bounds for certain variables. We have put this construction on Gomory's method to explain it in terms of the ideas developed in this paper, though the method evolved from a somewhat different set of notions.)

Both of the limitations we have indicated result from concessions to the pivoting rationale carried over from the Simplex algorithm, which requires the c vector to be altered at each step. Thus the method ignores the possibility that some synthesis of constraints other than the one it employs may occasionally be more desirable for expediting convergence. Such a possibility is not only meaningful in our algorithm, but is suggested by the method of Lemma 7, as we have already noted.

The specific way in which Gomory's method accomplishes the second stage of the pivoting process may be described as follows. The first step is to select an unsatisfied constraint J and carry out the row subtractions defined in step 3 of the example method in Section V. By applying these operations to the constraint $w_I \geq 0$, which is always available implicitly if not otherwise, an inequation is obtained that will have negative coefficients in every row where the Jth constraint was originally positive except, of course, in row I. Therefore, some nonnegative multiple of this inequation when added to the reduced Jth constraint will make a new constraint with all coefficients nonpositive except in row I. From the latter constraint a lower bound may be obtained for the new w_I, and the c vector adjusted accordingly. We present this procedure explicitly below.

<u>Gomory's All Integer Algorithm.</u>

1. Pick a positive component c_J of c. If no more exist the problem is solved.

2. Apply the row subtractions defined in step 3 of the example algorithm of Section V.

3. Carry out the transformations not only on the regular tableau matrix, but also on the adjoined constraint column corresponding to the inequation $w_I \geq 0$. Add the smallest nonnegative multiple of the resulting constraint to the reduced Jth column which will make a new constraint having exactly one positive coefficient.

4. Make all permissible row additions with respect to the new constraint, obtain the lower bound for the new w_I, adjust

the c vector, and return to 1.

From this it can more clearly be seen where the two algorithms differ. We will now hazard a guess or two about what may be expected from this difference. By using a predetermined simple rule for reducing an unsatisfied constraint, Gomory's algorithm should be faster (provided we do not use an equally simple rule) on problems which both methods find relatively easy to solve. For harder problems, the relative merits of the two methods will depend on a number of considerations. Generally speaking, the efficiency or inefficiency of our method as compared with Gomory's will depend on our ability to set up bounding forms, either by a single-minded strategy of applying elemental transformations or, as suggested in Section V, by combining such a strategy with a technique for creating new constraints. At the extreme, by eliminating the major part of choice in both dimensions, our method would become quite similar to Gomory's, except that we would continue to rely on the bound escalation method instead of obtaining lower bounds by dealing with one constraint at a time. This would be an advantage when appropriate bounding forms were encountered, but the cost of trying to create and then identify the bounding forms would constitute a comparable disadvantage if in fact it was rarely possible to find one consisting of more than a single column. Along the same lines, it must be pointed out that the freedom of choice available in our method may be a liability as well as an asset. This will certainly be true if no specific approach can be developed which works well [illegible] on a handful of problems, since the [illegible] of the remaining problems would then be delayed through irrelevant assessment. The only real answer to these issues must of course come from empirical results. We are presently programming our method for the Bendix G-20, and hope to have some experience with practical problems to report in the near future.

REFERENCES

1. DANTZIG, G. B., FORD, JR., L. R., and FULKERSON, D. R., "A Primal-Dual Algorithm for Linear Programs," in Annals Study 38 Linear Inequalities and Related Systems, Kuhn and Tucker, eds., Princeton University Press, 1956.

2. GLOVER, FRED, "A Study of the All-Integer Integer Programming Algorithm," O. N. R. Research Memorandum No. 110, Carnegie Institute of Technology, 1963.

3. GOMORY, RALPH E., "All-Integer Integer Programming Algorithm," IBM Research Report RC-189, International Business Machines Corporation Research Center, Yorktown Heights, 1960.